



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application)	PATENT APPLICATION
)	
Inventor: Multer et al.)	
)	Art Unit: 2175
Application No.: 09/753,643)	
)	Examiner: Mofiz, Apu M.
Filed: January 2, 2001)	
)	Customer No. 28554
Title: SYNCHRONIZATION SYSTEM)	
APPLICATION OBJECT INTERFACE)	
<hr/>		

**DECLARATION OF RICHARD M. ONYON
PURSUANT TO 37 C.F.R. §1.131**

I, RICHARD M. ONYON, declare that:

1. I am an inventor of the invention described and claimed in the above-identified patent application. I am currently the Chief Executive Officer of fusionOne, Inc., assignee of the instant application, and am a co-founder of the company. I have reviewed the pending application as stated in my Declaration for Patent Application and the pending claims as set forth in the RESPONSE A TO OFFICE ACTION ("RESPONSE A") accompanying this DECLARATION. I have also reviewed U.S. Patent No. 6,272,545 having a filing date of October 22, 1998, which patent claims priority as a continuation in part to United States Application No. 09/058,685, filed April 10, 1998, and which patent claims priority as a continuation in part to United States Application No. 09/058,528, filed April 10, 1998, and which application claims priority to United States Provisional Application No. 60/063,164, filed October 24, 1997, and which application claims priority to United States Provisional Patent Application No. 60/064,986, filed November 7, 1997.

2. I understand that this Declaration will be filed in the United States Patent and Trademark Office in order to provide factual evidence showing that the invention claimed in the present application was conceived of prior to the date of October 24, 1997.

3. The facts set forth hereinafter to establish that the claimed invention was conceived of prior to October 24, 1997 all relate to acts which occurred and were carried out within the United States.

4. In January of 1997, I began working for a company called Cachelink Corporation in Westboro, Massachusetts. Cachelink later changed its name to Mango Software. When I began working at Mango Software, the company was focused on developing a system for "pooling" the disk and memory resources of computers across a TCP/IP network. The product under development embodying this concept was called Medley97. Medley97 was a storage-oriented, distributed network operating system which pooled the resources of network-attached machines to create a single, virtual server.

5. I was hired at Mango Software as the Vice President of Sales and Marketing. My job was to transform Mango Software from an engineering only company into an operating company with sales and marketing functions. I was also charged with taking the Medley97 product to market.

6. Before the launch of Medley 97, and prior to October of 1997, I had a discussion with one of my former technical employees, Leighton Ridgard, about the technical merits of Medley97. Our discussions included how one could implement a system for keeping files current across a number of

machines which were not concurrently connected. Essentially, what I had envisioned was a synchronization solution, which I personally needed based upon my unrelenting travel schedule and the fact that I owned several computers. With the sync system I envisioned, if my home PC was always in sync with my office PC's files, I would not have to transport my work home at night. Or, if I was going on a business trip, I could simply take my laptop computer without concern for whether or not its contents were completely current. For example, even if I was in Japan, I could plug into the Internet and trust that my laptop's content would soon match my office PC's. While Mango focused on concurrent file access for groups of people across the Internet, I believed there was a bigger opportunity in keeping files "in sync" across multiple computers for one person.

7. The core concept conceived of during the conversation between myself and Mr. Ridgard described above, and which evolved over the months that followed, was originally focused on keeping data files, such as word processing, presentation and spreadsheet files, in sync. Syncing data files is a function which even today is not performed by a number of sync solutions. In general, I needed a system to keep files "concurrent" across a series of computers which were not located physically near each other in a way that was Internet centric (using, for example, TCP/IP), very efficient (so that even low bandwidth connections could be used to sync a PC) and asynchronous (since all PCs were never likely to be connected at the same time). At that time, all sync products of which I was aware were local in nature, and reconciled differences by comparing information on the two devices to be synchronized at the same time. Such a comparison requires a relatively high speed data connection, such as a physical cable between the devices. Our concept envisioned that a large file belonging to one user could be maintained in synchronization among two or more PC's on the

opposite ends of the country through a connection to the Internet because only a tiny "piece" of the file actually needed to be moved across the Internet when a change was detected.

8. In the months following the initial conception of my file sync idea, and prior to October, 1997, I began discussing the file sync concept with Sally Winship (now Sally Winship-Comollo, Vice President of Corporate Communications for fusionOne, Inc.), who was at that time Mango's press relations manager. During these discussions, Ms. Winship-Comollo and I also began discussing my desire to build a new company again. We agreed that the file sync product was a good idea and worthy of pursuit in a new business venture. My discussions with Ms. Winship-Comollo and Mr. Ridgard continued into the fall and winter of 1997/1998 through construction of the prototypes, and intensified as technical issues with Medly97 led to turmoil at Mango Software.

9. Prior to October 1997, Leighton Ridgard and I discussed how to synchronize Outlook data between two PCs. Initially we considered syncing Outlook's entire PST from one place to another, following the idea of syncing a data file. While this would have worked, it did not take us long to realize that Outlook's PST was large and sometimes only a small thing changed such as marking an email as read. Leighton knew of the existence of technologies to compute and efficiently generate the binary difference between two files. Xdelta was one such technology. We quickly decided that we would use binary differencing technology to generate a small difference between the PST files and transmit that difference during the sync. We began investigating acquiring binary differencing technology through purchase or writing it ourselves.

10. It did not take us long to further conclude that syncing PST with binary differences would only work when one wanted both machines to be identical. We quickly realized that in order to have one machine contain a subset of the data found on another machine we would have to parse the PSTs and extract only the required data. We decided at this point that Binary differencing would work for files. Outlook and other PIM content would need a different methodology.

11. Well prior to October 1997, I had several meetings with Ms. Winship-Comollo where we flushed out further details of the file sync system design. During at least one of these meetings, using the white board in her office, I drew block diagrams of the “store and forward” system. According to my original conception of a store and forward system, when a change is made to file data on a first computer, those changes would be recorded and transported via a network to be stored on an FTP server. At some later point in time, whether or not the first computer was coupled to the network, those changes could be forwarded to a second computer. In this conception, I envisioned some triggering event to forward and retrieve changes to and from the sync server. These drawings were erased from her whiteboard at the conclusion of our meeting and no copies of these drawings were made.

12. During our discussions regarding transmitting only that piece of the file which had changed, we realized that the method of creating and recording changes to only that portion of the data which had changed meant we needed some representation of the data file prior to the change. This need for a known state of a data file evolved into the so-called application object store or “AOS”. The AOS is essentially a known state of a user’s data as of the last time our system checks to determine whether any changes are made to the user’s data. The construction of the prototype of the AOS

application began prior to October of 1997, with numerous substantive technical discussions regarding the application taking place prior to October 1997.

13. It was at this point, again prior to October 1997, that we invented our change log and AOS technology. It was decided that we would need to extract from the PIM each item individually and transmit that item efficiently across to the other device. In order to do this we would need to compute the difference between the PIM data now and the PIM data at the time of last sync. Having already spent time examining delta technology (which we decided we would use for unstructured data such as files) we then created what we called "Structured Delta". Two systems of discovering the "difference" between the PIM now and when we last synced it was created. One was to monitor the PIM for user activity (changing, records, adding records etc.). Another was to store a copy of the PIM data at the end of the sync (in our AOS) and then to compare the stored copy with the PIM copy and determine the differences. Regardless of the method used to compute the differences, we created a system of representing the differences by using a set of operations followed by the data used by that operation. In other words, we created a log of activity (also called a "change log" or "data package") that contained instructions (such as add, delete, modify, rename, and move) along with the relevant data.

14. When creating the change log, we realized we should provide information in a header for the change log to describe the contents of the change log. At this point in time it was decided that the header should contain a signature, the identity of the generating device, the date, the PIM used, version schema in use etc.

15. Still well prior to October 1997, we realized that someone who syncs frequently would generate a lot of these change logs. At this time, we thought they would be stored on an FTP site, and storage space was a consideration. We further discovered that even after items were deleted their previous existence would be evident in the old change logs. In order to more efficiently manage the users space it was decided that a process could be developed to combine the change logs. We called this process "log rolling" at the time. Initially we thought about running this process on a back-end server without even involving the user. Such a process would apply successive change logs to each other and combine changes and deletes etc. to produce a minimal set of resultant changes collapsed into ideally one change log. During these discussions we decided that even if not done behind the scenes on the server, this feature would be equally valuable if implemented on the client. We later decided to called server-side log rolling "Base rolling" and left the term "log rolling" for use in describing client-side rolling.

16. Initially we considered syncing Outlook's entire PST from one place to another, following the idea of syncing a data file. While this would have worked, it did not take us long to realize that Outlook's PST was large and sometimes only a small thing changed such as marking an email as read. Leighton knew of the existence of technologies to compute and efficiently generate the binary difference between two files. Xdelta was one such technology. We quickly decided that we would use binary differencing technology to generate a small difference between the PST files and transmit that difference during the sync. To accomplish this, information contained within an application was compared against a store of that information as it existed at a point just after the previous synchronization of that application. This comparison is performed by an engine, which we referred to as the delta module.

17. As indicated in paragraph 9 above, prior to October 1997, Leighton Ridgard and I discussed specifically how to synchronize Microsoft Outlook data between two PCs. As we envisioned working with data and information from different vendor-unique applications in addition to Outlook, it was our thought to first translate content from the respective proprietary applications into a generic and universal format which could then be used by the delta module and other segments of the sync system. The initial discussions on this, all prior to October 1997, related primarily to working with Microsoft Outlook, and envisioned an "Outlook pipe" which was an application object specific for use with Outlook, and used to interface with Outlook to translate the Outlook content used by the delta module into a universal format. It was also envisioned at that time that the sync system should work with other vendor-unique applications to obtain and sync information, and that separate application object pipes would be provided to interface with and translate content from each of these different proprietary applications into a universal format.

18. Prior to October of 1997, we considered various configurations for the application object pipes used to translate content from the various vendor-unique applications into a universal format. One possible configuration we contemplated was an application object including a root object for providing an entry point into the vendor-specific application, and an interface object in the form of standard application programming interfaces (APIs) which could be used to gain access to the application content. We also considered that each application object could include a subset of child objects into which various types and classes of information from the vendor-unique application could be mapped. Each of these features of the application object pipes was conceived of prior to October of 1997.

19. Our discussions on finalizing the structure proceeded rapidly and within two weeks of when we started we had a thorough understanding of what we wanted to build.


20. After this initial series of technical discussions, I asked Mr. Ridgard to construct a prototype of the system based on these discussions. I recall this construction beginning prior to October of 1997 and being completed in or about February 1998. I am aware that his recollection is that he began later. Despite my eagerness to get started, Leighton could not begin until then because he was employed elsewhere and had other responsibilities. In the interest of time (and because he was the only person working on this) it was decided to produce a proof of concept prototype which synced the entire PST from one device to the other. Although this was certainly not the implementation upon which we had decided, it was sufficient as a proof of concept. In the spring of 1998 two other engineers joined him and we produced the second prototype which implemented an early version of our change log technology.

21. Throughout our development of our system, we worked diligently to reduce the invention to practice. This included the creation of at least three (3) prototypes throughout the process of hiring engineers and gaining funding to complete development of the system.

22. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by

fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Date: January 2004

By: 
Richard M. Onyon